3D Feature Surface Properties and Their Application in Geovisualization

Haik Lorenz, Jürgen Döllner

Hasso-Plattner-Institut, University of Potsdam, August-Bebel-Str. 88, 14482 Potsdam, Germany

Abstract

New acquisition methods have increased the availability of surface property data that capture location-dependent data on feature surfaces. However, these data are not supported as fully in the geovisualization of the Digital City as established data categories such as feature attributes, 2D rasters, or geometry. Consequently, 3D surface properties are largely excluded from the information extraction and knowledge creation process of geovisualization despite their potential for being an effective tool in many such tasks. To overcome this situation, this paper examines the benefits of a better integration into geovisualization systems in terms of two examples and discusses technological foundations for surface property support. The main contribution is the identification of computer graphics techniques as a suitable basis for such support. This way, the processing of surface property data fits well into existing visualization systems. This finding is demonstrated through an interactive prototypic visualization system that extends an existing system with surface property support. While this prototype concentrates on technology and neglects user-related and task-related aspects, the paper includes a discussion on challenges for making surface properties accessible to a wider audience.

Key words:
geovisualization, exploratory data analysis, 3D surface properties, textures, computer graphics, GPU

Email addresses: haik.lorenz@hpi.uni-potsdam.de (Haik Lorenz), doellner@hpi.uni-potsdam.de (Jürgen Döllner)

1. Introduction

Today's geovisualization systems enable users to work with data from various basic categories, such as feature attributes, 2D rasters, or multidimensional geometry. These categories fundamentally differ with respect to domain, usage, and technology. Each phenomenon can be represented in a Digital City using various data types depending upon performance, precision, the availability of algorithms or system support, or the availability of data or capabilities for its capture. Geovisualization systems provide users with category-specific means for transforming, correlating, and visualizing all data with the goal of extracting information and knowledge. Highly developed systems facilitate *flexible visualization* (Dykes, 2005) that encourages ideation in a non-predefined ways and enables *exploratory data analysis* (EDA).

Recently, an additional data category has received increasing attention: *3D surface properties*. This category describes data attached to 3D feature surfaces; i.e., for each surface location, a data value of an arbitrary, but homogeneous, type can be stored. Surface properties are not limited to the Earth's surface but apply to any 3D feature, such as buildings or bridges. The simplest surface property is a constant value assigned to the feature's entire surface. Usually, surface properties provide location-dependent, varying values. Typically, such a property is sampled regularly and stored as a collection of 2D rasters along with one unique mapping function per surface patch (e.g., a polygon). A surface property covers the complete surface and maps a unique raster portion to each surface patch (Figure 1). Some surface properties can be described procedurally; i.e., there is a function that computes the property value given the surface location. This applies, for example, to surface-based simulations, even though the computational efforts for on-demand evaluation may be prohibitively high. Surface properties fit into the well-established coverages concept (Open Geospatial Consortium, 2000). The difference between surface properties and existing coverage implementations, such as (massive) georeferenced 2D rasters, is the highly fragmented spatial domain.

New techniques enable the cost-effective acquisition of surface properties even for massive data sets. Examples are realistic and fully textured virtual 3D cities available in Google Earth or Bing Maps, such as Munich (Germany), Zurich (Switzerland), Manhattan (NY, USA), and others. There are various sources capable of producing data in this category:

- Remote sensing, e.g., facade textures from oblique imagery (Früh et al.,

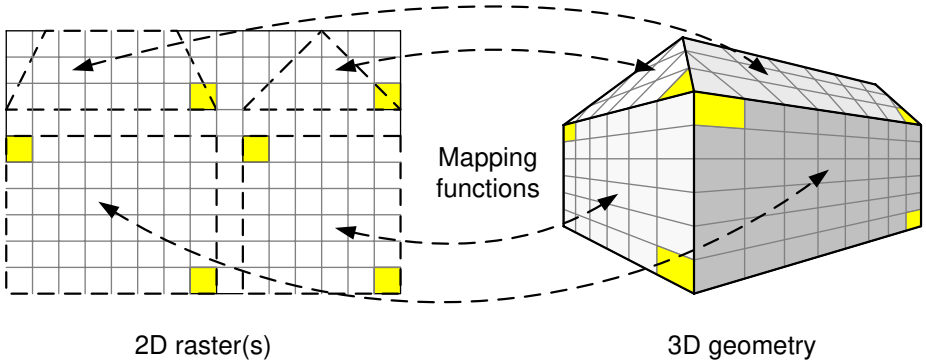

Figure 1: Schematic view of a raster-based surface property. Each surface patch is mapped to an associated unique raster area.

2004; Lorenz and Döllner, 2006),

- Physical simulation, e.g., photovoltaic potential (Šúri and Hofierka, 2004) or radio network coverage (Maciel et al., 1993),

- Spatial analysis, e.g., solar envelopes (Morello and Ratti, 2009), or architectural metrics such as vista quality, or building facade visibility.

Increasing data availability has been acknowledged with the inclusion of 3D surface properties in CityGML (Gröger et al., 2008), the international standard of the Open Geospatial Consortium (OGC) for exchanging and storing virtual 3D city models. Naturally, surface properties should also be part of the information and knowledge extraction process (Gahegan, 2005) that drives geovisualization. Some potential benefits of integrating surface properties include the following:

- More precise phenomenon representation, e.g., finer granularity, or less abstraction of surface-related phenomena.

- Improved data access, e.g., the direct usage of surface properties generated by special-purpose software in geovisualization systems without the need for conversion to other data categories.

- Improved analysis techniques, e.g., access to more precise data, access to additional data, or incorporation of more complex analytical models.

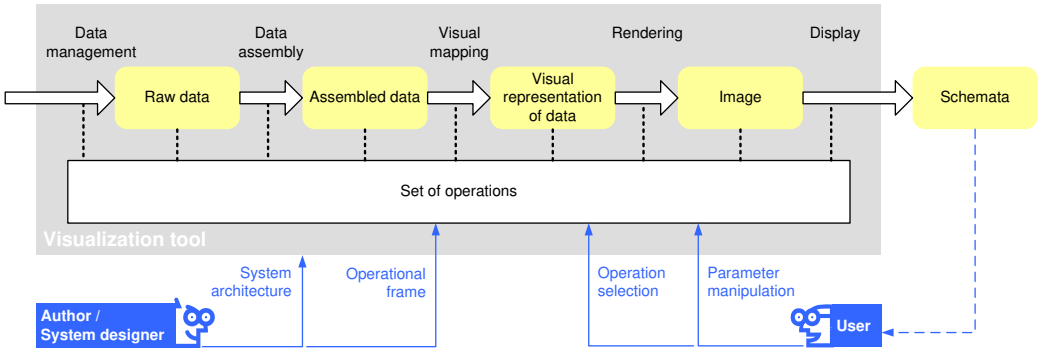

Figure 2: The visualization pipeline (modified from Wood et al. (2005) and Spence (2000)).

For successful integration into geovisualization and EDA, systems must provide support in terms of specialized means for transforming, correlating, and visualizing surface properties. However, geovisualization systems rarely support this data category. Some systems provide very limited capabilities in terms of 2D textures being applied to 3D geometry solely for display purposes. This approach assigns a passive role to surface properties and is insufficient for enabling their effective use. This paper aims to improve the support for surface properties and thus has two major objectives. First, we present reasons for their integration into geovisualization systems, and second, we discuss relevant technical foundations. We demonstrate our approach through a highly interactive prototype.

What is sufficient system support? Visualization systems enable users to adapt, or interact with, a visualization within bounds determined by the author and the system itself. These systems are conceptually based on the *visualization pipeline*, which describes the data flow from raw data to final visualization. This paper uses the pipeline presented in Wood et al. (2005), which is a variation of the original Upson/Haber-McNabb model. This pipeline consists of five stages (Figure 2):

1. Data management
2. Data assembly
3. Visual mapping
4. Rendering
5. Display

According to interaction categorizations for information visualization (Chi and Riedl, 1998) or cartographic visualization (Persson et al., 2006), all in-

teraction facilities or *operations* available in a system can be assigned to one of these stages or to transformations between consecutive stages. From a technical point of view, interaction is the selection of operations and the manipulation of parameters. From the user's point of view, interaction is the result of a mapping of an intent or task to accessible system capabilities (Yi et al., 2007). For sufficient support, visualization systems must provide access to a set of operations, which is either focused on given tasks (Andrienko et al., 2005) or comprehensive for flexible visualization. As a result, users should be able to reach their goals. Of the two main aspects of support, namely, access to operations (that is, the user interface) and the operations themselves, this paper focuses on operations.

For well-established data categories of Digital City representations, such as feature attributes, 2D raster data, or 2D/3D geometry, a wide selection of operations has been described in the literature and/or exists in both scientific and commercial systems. Examples can be found in many standards issued by the Open Geospatial Consortium (OGC). They include the Web Feature Service (Data management; Vretanos, 2005b), Filters (Data assembly; Vretanos, 2005a), and Symbology Encoding (Visual mapping; Müller, 2005). Other operations include 3D navigation (Rendering) and 3D printing (Display). An important aspect is merging data from different categories, e.g., selecting 2D lines based on an associated attribute.

Surface properties require a similarly rich set of operations. In this paper, we discuss the usage of surface properties with respect to other data categories and identify basic properties. Based on these properties, we discuss similar concepts from the field of computer graphics and investigate their applicability in a geovisualization environment. As proof of concept, we describe a prototypic surface property visualization application, which provides a highly flexible visual mapping. This application is meant to be a test bed for technology, not a system available to a wide audience. It provides only a minimalistic user interface that does not shield users from the complexities of the underlying technology. Starting from this proof of concept, we discuss technological challenges in completing surface property support in order to make it an accessible tool for users.

The remainder of this paper is structured as follows. Section 2 provides two usage scenarios that exemplify the benefits of a tight integration. Section 3 discusses surface property operations and their relation to programmable graphics cards and provides implementation details of our prototypic geovisualization tool. Section 4 outlines challenges for a more complete integration

of surface properties into geovisualization systems. The paper concludes with an outlook in Section 5.

2. Example Usage Scenarios

Two usage scenarios highlight the benefits of a tight integration of surface properties into the visualization pipeline.

2.1. Photovoltaic Potential Analysis and Visualization

In recent years, renewable energy has become a major topic of interest. Photovoltaics (PV) represent an important technology for building owners, as the technology is becoming cheaper. Cost efficiency depends on various parameters, such as annual insolation, PV panel area, panel orientation, and occlusion, that is, the percentage of light actually reaching the surface that is not obstructed. Simulations can determine cost efficiency for a given surface.

There are commercial providers that determine cost efficiency for single buildings. They provide a report with a given set of analyses, i.e., a non-interactive visualization. Alternatively, an increasing number of municipalities build so-called *solar cadastres*. The project SUN-AREA (Ludwig et al., 2008) relies on laser scanning to acquire relevant roof and occlusion information. It then uses a simulation to determine the PV potential of each roof surface based on assumptions for panel efficiency. The results are a GIS and a web-based interactive 2D map (Figure 3) that help building owners reach a decision.

A shortcoming of many current solar cadastres is the exclusion of walls, even though they can be suitable for, and should be considered as, candidate locations as well. Without surface properties, the PV potential of walls is hard to represent. With surface properties, its calculation becomes straightforward. Moreover, surface properties are already used in PV simulation algorithms. First, occlusion is represented as a surface property varying across the surface. Second, PV panels must be fitted onto potential surfaces such that non-usable areas like windows remain uncovered. Non-usable areas can be stored as masks in a surface property. Finally, detailed PV simulation inherently produces surface properties.

Without visualization tool support, surface properties must be summarized in terms of attribute values before visualization. Retaining surface properties for visualization instead increases the degree of detail in visualization dramatically. If, in addition, surface properties become as flexible as

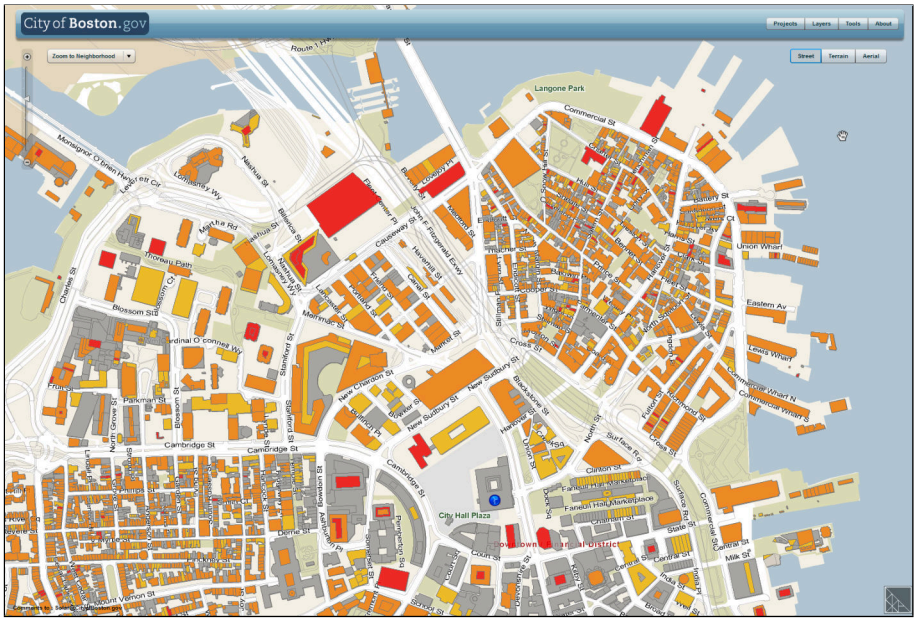

Figure 3: Example of a 2D photovoltaic potential visualization. Solar Boston (City of Boston, 2009) provides a citywide PV potential analysis as a 2D WebGIS application. The figure shows the total annual solar radiation available at the rooftop with red hues indicating higher and yellow hues indicating lower availability. Source: The Boston Redevelopment Authority, Solar Boston Program, Office of Digital Cartography and GIS (ODC&GIS), 2009, Boston, MA, USA.

other object types, new interactive analyses become possible. For example, the minimum acceptable effective insolation can be varied interactively to compare different PV installations. If merged with attribute data, analyses can be focused on buildings of specific type or usage. These analyses do not require the original PV simulation software but are carried out solely in the visualization system and thus are available to a much larger audience. Also, the simulation software does not need to provide a comprehensive visualization component, as users can rely on their favorite visualization system for exploration and further analysis, as long as it supports surface properties.

2.2. Assessment of Residential Quality

Residential quality has become an important criterion for home seekers, real estate agents, architects, and urban planners. However, it does not have an objective definition. Instead, there are a large number of factors – both

objective and subjective – that are supposed to have an influence on residential quality, including distance to public transport or schools, noise pollution, light exposition, and the visibility of vegetation and city silhouettes.

Assessing residential quality requires weighting all factors in a subjective way. A visualization tool may be an appropriate means of enabling those making decisions to customize these weights (Williamson and Shneiderman, 1992). The resulting visualization enables them to explore residential quality and compare different weightings. Without surface properties, assessment granularity is bound to individual objects, i.e., mostly buildings, or to the Earth's projected surface. The results are presented either in a virtual 3D city model or as a 2D map. However, some applications require a finer granularity, e.g., per apartment or even per room. Established data categories cannot provide this granularity. Surface properties are capable of storing room or apartment attributes for a whole building by assigning each room's or apartment's attribute value to the respective surface part. The result is a discrete, piecewise constant surface property. Moreover, some important phenomena, such as noise pollution, light exposition, and the visibility of vegetation, depend on 3D positions and vary largely across building facades in a continuous fashion. Hence, they must be represented by surface properties for reliable results.

In terms of the visualization pipeline, relevant operations for the assessment task mainly belong to the data assembly and visual mapping stage. For data stored as feature attributes and georeferenced 2D rasters, such operations are well-defined, e.g., by the OGC Filter standard (Vretanos, 2005a). The overall approach to the residential quality assessment should not change if surface properties are incorporated. Thus, comparable data assembly and visual mapping operations are required for all data categories, including surface properties.

A visualization system with basic surface property support visualizes the resulting residential quality in terms of color-coded layers on all building surfaces (Figure 4). Users are enabled to interactively adjust weightings but need to actively explore the virtual 3D city model to find suitable places. The accompanying video showcases this approach (Figure 4). Visualization systems with more advanced support could evaluate the resulting surface property and present users with a list of best places, and since the 3D locations of the places are known, it could draw attention to them with appropriate visual encodings and provide automatic navigation.

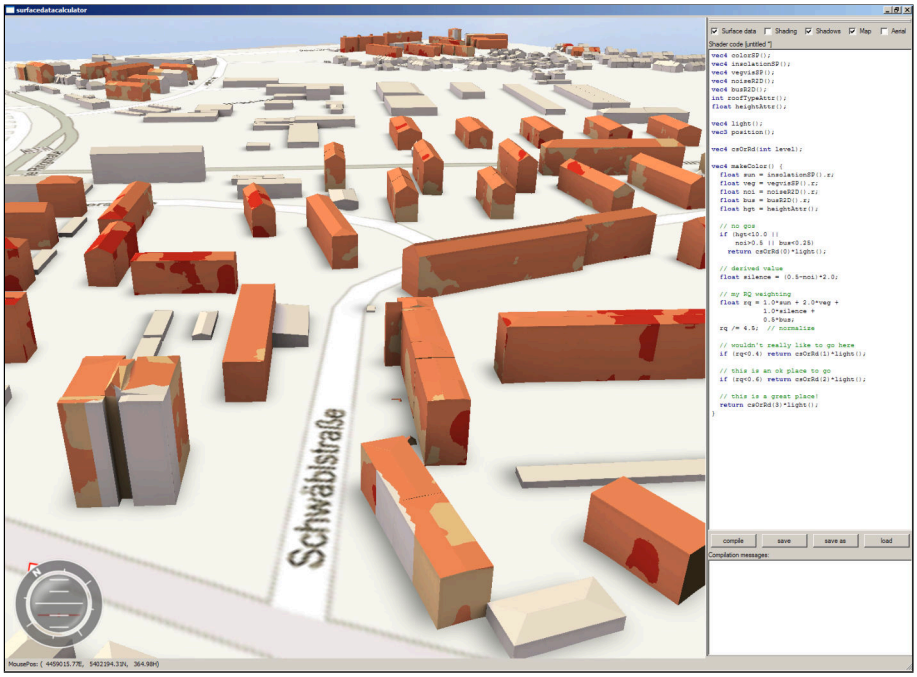

Figure 4: Residential quality visualization using surface properties. Users can create their own weighting and color mapping from all available data sources using small code fragments. Here, red tones denote suitable places and light orange tones unsuitable places. The accompanying video shows how this approach may be used.

3. Operations for Surface Properties

Operations are the functional core of interactive geovisualization systems. A surface property operation applies to an entire covered surface and is conceptually executed once for each cell in each raster of the resulting surface property. Due to the complex structure of surface properties, three different categories must be distinguished:

Location-based operations: For each cell, the corresponding 3D location must be known, e.g., distance calculations to a set of points.

Patch-based operations: For each cell, only the corresponding surface patch must be known, e.g., the incorporation of attributes of the feature to which the surface patch belongs.

Property-based operations: The operation does not refer to surface

patches or 3D locations, e.g., the addition or multiplication of whole surface properties.

In addition, surface properties can be used together with data from other categories. There are two usage scenarios:

1. Surface property operations can use other data categories as input. Conceptually, this requires converting the various categories to surface properties. Feature attributes apply to an entire feature and thus are constant for its respective surface. 2D rasters are usually georeferenced. For a given surface location, the corresponding value can be determined by projection onto the raster's reference plane. These two conversions are employed in the residential quality example (Section 2.2). Geometry handling is more complex and usually relies on rasterization or ray tracing. For example, occlusion estimation, which is part of the PV analysis example (Section 2.1), can use shadowing algorithms to generate an occlusion surface property from feature geometry.
2. Surface properties can be used by operations on other data categories. For this, they must be reduced to either feature attributes or 2D rasters. Reduction to attributes requires the aggregation of all surface property values within a given surface to a single value, usually through statistical functions, such as *min*, *max*, or *average*. Reduction to 2D rasters requires projection onto the raster's reference surface.

The concept of surface properties is tightly related to the notion of 2D texture mapping or, more specifically, unique 2D texture mapping (Lévy et al., 2002) in computer graphics. Textures are not just image data but rather are regarded as generic 2D data containers, which makes them a perfect vehicle for implementing flexible surface properties. In addition, so-called shaders (Akaine-Möller et al., 2008) allow for the application of arbitrary functions to each pixel with built-in texture support and thus can serve as a framework for the aforementioned operations.

For the implementation of efficient surface property operations, we can rely on existing out-of-core rendering strategies, as the amount of data usually exceeds the available graphics memory. A number of techniques are available, such as view-dependent texture atlases (Buchholz and Döllner, 2005; Cignoni et al., 2007), clipmaps (Tanner et al., 1998), and virtual textures (Mittring, 2008; Barrett, 2008). All techniques share the idea of a texture budget that is filled only with relevant parts of the texture data. Whenever new data

becomes relevant, it is loaded from storage into graphics memory, thereby overwriting currently irrelevant data.

Operations can easily access surface property data if implemented in shaders, since texture access is a core feature of shaders and includes a wide variety of mapping and filtering options. Shaders can also access non-texture data such as attributes. As shaders are executed on graphics hardware, the application itself must load all required data into graphics memory. Shader output is a color or numerical value. Shaders are written in high-level programming languages similar to C and impose few limitations on implementable algorithms. The most important limitation is separation; the shader code is executed or invoked once for each output value in a massively parallel fashion. That is, all invocations conceptually run independently at the same time, prohibiting the exchange of information, e.g., for saving redundant computations. If operations require such an exchange, they must be split into parts where each part produces a complete intermediate surface property, which then serves as input for the next part. A consequence of this separation is the limited ability to perform aggregation operations for entire surface patches or features. Each shader invocation would need to read the entire surface property and calculate the result itself, thereby performing highly redundant computations. It is much more efficient to pre-compute such operations outside the shader and only provide the result for further use (Section 3.2).

To provide proof for the applicability of computer graphics techniques, we implement a software prototype. Chi and Riedl (1998) propose three different implementation choices for operations: (1) inside the visualization system, (2) inside a data management system (DBMS), and (3) in a separate analytical engine. All three choices can make use of computer graphics technology. We opt for the visualization system because it is most accessible to us and already contains a real-time computer graphics environment for interactive visualizations as a basis for our implementation. The described system concentrates on the technological framework, rather than on usability or specific tasks, and consequently does little to hide the underlying graphics concepts. In other words, if surface properties are to be made available to a wider audience, they need to be integrated into standard geovisualization or geoinformation systems. Such an integration requires the exploration of the remaining two implementation options as well as the design of methods to access surface property operations through more familiar interfaces such as toolboxes, queries, or dialogs. Such issues are discussed in Section 4.

3.1. Implementation

Our prototypic visualization tool concentrates on the visual mapping of surface properties in virtual 3D city models that contains all data categories. It provides basic operations for the visual exploration of 3D city models, such as the loading of data sets or 3D navigation. It does not contain fixed surface property operations but provides direct access to the shader code for users to implement application- and task-specific operations. A major aspect is the merging of surface properties with other data categories. For this aspect, all available data are provided to the shader. The incorporation of surface properties into operations with respect to other data categories has not yet been implemented. Aggregation operations are also not implemented yet. Both are discussed in Section 4.

The tool uses Autodesk LandXplorer (Autodesk Inc., 2009), the Virtual Rendering System (VRS; Döllner and Hinrichs, 2002), and OpenGL as graphics API. Shaders are written in GLSL (Rost, 2006), the OpenGL Shading Language. A key functionality is providing all data contained in the virtual 3D city model to the shader. The different data categories are treated as follows:

Feature Attributes: Feature attributes of acceptable type are collected into arrays that are then copied to graphics memory. Acceptable types include integers, floating point numbers, booleans, and vectors. Strings currently cannot be processed in a straightforward manner in GLSL and are excluded.

2D Rasters: 2D rasters are directly stored in textures if they do not exceed a hardware-dependent dimension limit. If they do, a texturing technique for massive texture data (Döllner and Baumann, 2000) is used.

2D and 3D Geometry: Geometry is used for rendering and requires tessellation of surface patches into triangles. Each triangle is augmented with a patch ID and a feature ID that can be used to load respective attributes during shader execution. Additionally, mapping functions for each surface property and 2D raster are added.

Surface Properties: This prototype focuses on raster-based surface properties. They are considered textures and preprocessed for out-of-core rendering based on the massive texturing technology available in the

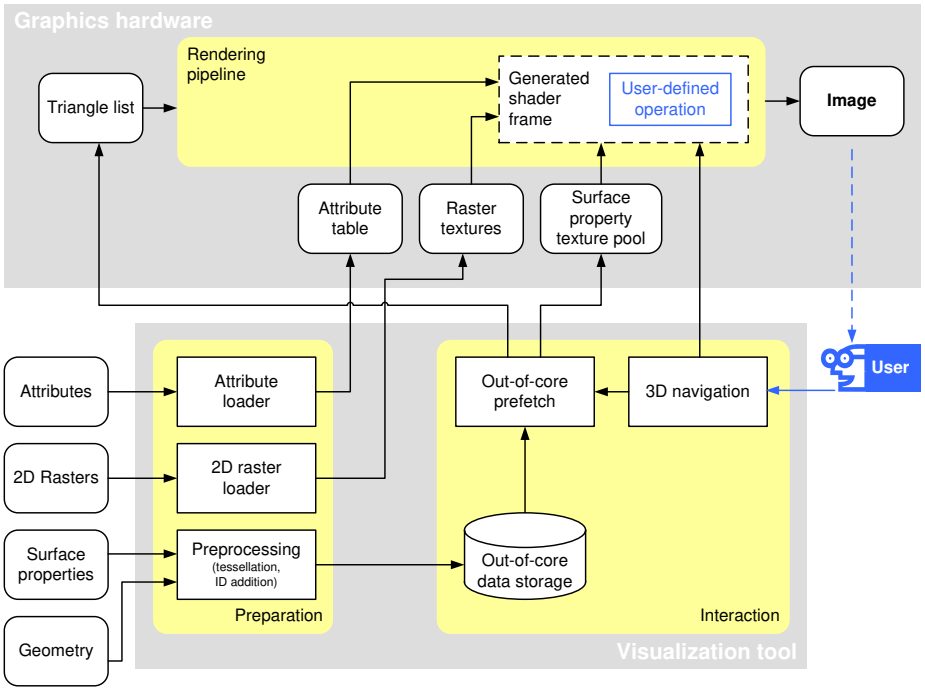

Figure 5: Data flow in our visualization tool with surface property support.

LandXplorer library. During rendering, the out-of-core technique ensures the availability of required surface property data.

After loading all data, shader code is generated that provides easy access to all data such that the prototype can be quickly configured in real-time with operation code. Figure 5 shows the application architecture and data flow. Within the shader, additional properties such as local lighting or 3D location are made available. The specification of operations through shader code is demonstrated in the accompanying video (Figure 4).

During rendering, the shader is executed for each pixel in which a surface is visible; i.e., operations are not applied to all cells available in a surface property but to all contributing pixels in the current view. This typically reduces the number of shader invocations and enables interactive frame rates. A disadvantage of this approach is aliasing, since surface property resolution(s) and pixel resolution do not match in most views. Advanced texture filtering capabilities and anti-aliasing through multi-sampling help reduce this problem. Both features are provided by the graphics hardware and have

no effect on the shader code. Users may merely notice some slowdown during rendering.

The integration of surface properties does not require major changes to the original geovisualization system architecture. The most significant additions are the exposition of the shader to the user, which includes shader frame generation, and the provision of attribute data to the shader. The remaining architecture is taken directly from the Autodesk LandXplorer platform. Hence, our approach allows for surface properties to be added to existing geovisualization systems with minimal effort. Moreover, our system immediately benefits from technology for handling massive data sets and scales as well as the original visualization system. A less prototypic surface property support requires additional components for improved user interface and complex operations. We expect them to fit well into our architecture.

The performance of this architecture is mainly influenced by the data-loading rate. Rendering requires all necessary data to reside in graphics memory. Out-of-core algorithms hide load latencies by reverting to coarser resolutions (both for texture and geometry) until optimal data is available. This ensures interactive frame rates, even if data comes from a comparatively slow source such as a remote database. Typically, prefetching further reduces latencies. However, if large parts of the view change, e.g., due to rapid view point movements or dynamic texture data, latency hiding fails, and rendering must wait for data transfer from storage to graphics memory. Examples for dynamic texture data are time-varying sequences and simulation results with time-varying parameters. If no (or an insignificant amount of) data need to be transferred to graphics memory, rendering can fully utilize the GPU. In particular, dynamic processes such as distances to moving objects, which are computed on the GPU, have almost no influence on the frame rate.

3.2. Limitations

Some aspects of computer graphics technology impose limitations on our approach. The current programming model for graphics hardware imposes strict limitations on memory access. While random reads are possible, random writes are not. Each shader invocation can write to predefined memory locations only, which restricts the class of implementable algorithms. Such restrictions are softened from hardware generation to hardware generation but still require unusual approaches to the implementation of some operations. Effectively, current graphics hardware is limited to the raster part of surface properties, i.e., operations that have a raster of a surface property

as output. If surface properties are to be used in operations on other data categories, such as feature attributes, or if the reference geometry of a surface property is to be changed, the described architecture must be extended by an additional processor that resides in the visualization tool. Examples of such operations include object selection based on surface property values, which requires the computation of a single boolean value from the surface property data of an entire feature, and the identification of visually similar buildings, which requires the projection of the respective surface property data onto a common reference geometry.

4. Challenges for Surface Property Integration

While the implementation of surface property operations may rely on existing technology, there are many more challenges that must be addressed before sufficient surface property support is reached. In the following section, we present a non-exhaustive list of open questions for further research. We group this list by subject.

Operations The most important question in this subject involves which operations are appropriate for surface properties. Since surface properties relate to both geometry and rasters, both domains could be sources for potential operations. For example, geometric buffering operations might prove useful. Similarly, raster-based segmentation operations might apply to surface properties as well. In addition, new kinds of operations might emerge from the tight link between both domains in surface properties. Currently, the lack of well-defined complex and powerful operations hinders the effective use of surface properties beyond simple examples. An example of a complex operation is the transfer of a surface property to a new reference surface. Such an operation could be useful for processing surface properties with similar but not identical reference surfaces.

Similarly, effective ways of including surface properties into well established operations on other data types must be explored. For example, the request of all apartments within a 15 minute walk to a train station could additionally ask for a good vegetation visibility. If the vegetation visibility is provided as surface property, this property must be reduced to a binary answer that faithfully reflects the "good visibility" condition. Various approaches can be though of, such as minimum and average visibility. Criteria for the selection of a reduction approach must be determined.

As pointed out earlier, computer graphics already provide a large amount of technology for the implementation of surface property operations. However, some problems cannot be solved in a straightforward manner and require other means besides computer graphics. It is necessary to identify operations that cannot be implemented, can partly be implemented, and/or can be implemented but are inefficient using graphics technology. For these operations, alternative implementations must be developed.

Operation access Operation access has been excluded from the scope of this paper. Nevertheless, it is an important aspect for developing sufficient support. While direct access to shader code as used in our prototype provides high flexibility, it requires in-depth knowledge and programming skills by the user. Dykes (2005) identifies a broad range of choices for operation access, such as GUIs, scripting, and code libraries, that differ in skill requirements, flexibility, and the level of interactivity. All choices expose complex operations in terms of black boxes and hide the actual low level implementation. For sufficient support, many of these access methods must be covered, yet operation access is orthogonal to operations, since the same operation implementation can be exposed to the user through multiple interfaces.

Implementation frameworks From the three implementation options proposed by Chi and Riedl (1998), we have described the implementation within the visualization system. The other two options, namely, within a DBMS and within a separate analytical engine, can make use of computer graphics technology as well, which makes surface properties available to a much larger group of applications. This includes visualization tools that do not contain operations for surface properties themselves but are capable of displaying textured city models. Depending on the implementation choice, GIS and plain database tools can readily use surface properties for data analysis.

Operation implementation in a non-visualization environment does not require the full architecture of an interactive visualization system. For example, out-of-core algorithms are not needed. Instead, efficient streaming algorithms for transferring input data to and results from the graphics hardware are necessary. Such algorithms are well known in the GPGPU (general purpose programming on GPUs) community. If graphics hardware is not readily available, a suitable operation framework must be provided in

software; starting points could be image processing libraries or software 3D renderers.

Geovisualization Our prototype implementation does not incorporate more complex aspects of 3D geovisualization. For example, it disregards issues of scale (Butkiewicz et al., 2008). Distant objects receive a small amount of screen space, which may be too small for presenting information. For the task of change detection, Butkiewicz et al. (2008) propose the use of additive splatting to highlight changes. In other contexts, levels of detail or generalization (Chang et al., 2008) are used. A canonical solution has not been described yet. Similarly, scale-aware visualization with surface properties requires solutions, which are derived from the intentions and priorities of specific users.

The issue of scale is a special case of the general problem of surface property display. Geovisualization has developed effective strategies for visually communicating spatial information, which mostly rely on the degrees of freedom as described by Bertin's visual variables (Bertin, 1983). Surface properties differ from other data categories in their density and the feature surface they occupy. Both influence the availability of visual variables and may require the development of new visualizations.

5. Conclusions

Surface properties represent a generic, multi-functional data category for Digital City applications and systems. Whilst not novel in their own right, new acquisition methods such as automatic facade texture creation and generative methods such as 3D visual analysis and simulation have brought surface properties to the attention of a larger audience. This paper has demonstrated applications of surface properties for two scenarios, namely, photovoltaic potential analysis and residential quality assessment, which can be systematically designed and implemented based on surface property concepts. In general, surface properties have the potential for increasing expressiveness and detail in simulations and analyses as well as for capturing real-world phenomena using digital city models. To take advantage of their potential and to make effective use of them, surface property concepts can be flexibly adapted to given contexts and tasks.

To make effective use of surface properties in geovisualization and EDA, visualization systems and applications require efficient implementations of

surface properties. This paper has focused on the implementation of operations as part of the visualization pipeline. The paper has presented ways in which computer graphics technology can be used as a basis for implementation, most of all for geovisualization, but also in the context of more general geoinformation systems. In particular, the successful integration of surface properties into applications and systems does not require major changes to the software architecture but the addition of orthogonal functional components. Starting from this basis, other aspects must be explored including operation access, implementation frameworks, and visualization aspects.

In the long term, surface properties should be supported in geovisualization and geoinformation systems in ways comparable to the current support available for attributes, 2D rasters, and geometry. Most likely, surface properties will become a core functionality of future Digital City systems, as they appear to be a system feature that relates to generic and multi-purpose functionality.

Acknowledgements

We thank Autodesk Inc. for providing the LandXplorer system platform and virtualcitySYSTEMS GmbH for providing parts of the dataset. We are grateful for the comments from four anonymous reviewers, which led to significant improvements of the manuscript, and wish to acknowledge contributions by members of the ICA Commission on GeoVisualization through their GeoViz Hamburg meeting. This work has been funded by the German Federal Ministry of Education and Research (BMBF) as part of the InnoProfile research group "3D Geoinformation" (www.3dgi.de).

References

Akaine-Möller, T., Haines, E., Hoffman, N., January 2008. Real-Time Rendering (Third Edition). A K Peters.

Andrienko, G., Andrienko, N., Dykes, J., Mountain, D., Noy, P., Gahegan, M., Roberts, J. C., Rodgers, P., Theus, M., 2005. Creating Instruments for Ideation: Software Approaches to Geovisualization. In: Dykes, J., MacEachren, A., Kraak, M.-J. (Eds.), Exploring Geovisualization. Elsevier, London, Ch. 5, pp. 103–125.

Autodesk Inc., 2009. Landxplorer.
http://www.autodesk.com/landxplorer.

Barrett, S., 2008. Sparse Virtual Textures.
http://silverspaceship.com/src/svt/.

Bertin, J., 1983. Semiology of graphics. University of Wisconsin Press.

Buchholz, H., Döllner, J., 2005. View-Dependent Rendering of Multiresolution Texture-Atlases. In: Proceedings of the IEEE Visualization 2005. IEEE, pp. 215–222.

Butkiewicz, T., Chang, R., Wartell, Z., Ribarsky, W., 2008. Visual analysis and semantic exploration of urban lidar change detection. Comput. Graph. Forum 27 (3), 903–910.

Chang, R., Butkiewicz, T., Ziemkiewicz, C., Wartell, Z., Pollard, N., Ribarsky, W., 2008. Legible simplification of textured urban models. IEEE Comput. Graph. Appl. 28 (3), 27–36.

Chi, E. H.-h., Riedl, J., 1998. An Operator Interaction Framework for Visualization Systems. In: INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization. IEEE Computer Society, Washington, DC, USA, pp. 63–70.

Cignoni, P., Di Benedetto, M., Ganovelli, F., Gobbetti, E., Marton, F., Scopigno, R., Sept. 2007. Ray-Casted BlockMaps for Large Urban Models Streaming and Visualization. Computer Graphics Forum 26 (3), 405–413.

City of Boston, 2009. Solar Boston. http://gis.cityofboston.gov/solarboston/.

Döllner, J., Baumann, K., 2000. Texturing Techniques for Terrain Visualization . In: Proceedings of IEEE Visualization. pp. 227–234.

Döllner, J., Hinrichs, K., 0 2002. A Generic Rendering System . IEEE Transactions on Visualization and Computer Graphics 8 (2), 99–118.

Dykes, J., 2005. Facilitating Interaction for Geovisualization. In: Dykes, J., MacEachren, A., Kraak, M.-J. (Eds.), Exploring Geovisualization. Elsevier, London, Ch. 13, pp. 265–291.

Früh, C., Sammon, R., Zakhor, A., 2004. Automated Texture Mapping of 3D City Models With Oblique Aerial Imagery. In: 3DPVT. IEEE Computer Society, pp. 396–403.

Gahegan, M., 2005. Beyond Tools: Visual Support for the Entire Process of GIScience. In: Dykes, J., MacEachren, A., Kraak, M.-J. (Eds.), Exploring Geovisualization. Elsevier, London, Ch. 4, pp. 83–99.

Gröger, G., Kolbe, T. H., Czerwinski, A., Nagel, C. (Eds.), 2008. OpenGIS City Geography Markup Language (CityGML) Implementation Specification. Version 1.0.0, doc.no. 08-007, OGC.

Lévy, B., Petitjean, S., Ray, N., Maillot, J., 2002. Least squares conformal maps for automatic texture atlas generation. ACM Trans. Graph. 21 (3), 362–371.

Lorenz, H., Döllner, J., 2006. Towards Automating the Generation of Facade Textures of Virtual City Models . ISPRS Commission II, WG II/5 Workshop, Vienna.

Ludwig, D., Klärle, M., Lanig, S., 2008. Automatisierte Standortanalyse für die Solarnutzung auf Dachflächen über hochaufgelöste Laserscanningdaten . In: Strobl, J., Blaschke, T., Griesebner, G. (Eds.), Angewandte Geoinformatik 2008 - Beiträge zum 20. AGIT Symposium. Wichmann, pp. 466–475.

Maciel, L. R., Bertoni, H. L., Xia, H. N., 1993. Unified approach to prediction of propagation over buildings for all ranges of base station antenna height. IEEE Transactions on Vehicular Technology 42 (1), 41–45.

Mittring, M., 2008. Advanced virtual texture topics. In: SIGGRAPH '08: ACM SIGGRAPH 2008 classes. ACM, New York, NY, USA, pp. 23–51.

Morello, E., Ratti, C., 2009. Sunscapes: 'Solar envelopes' and the analysis of urban DEMs. Computers, Environment and Urban Systems 33 (1), 26–34.

Müller, M. (Ed.), 2005. Symbology Encoding Implementation Specification. Version 1.1.0 (draft), doc.no. 05-077, OGC.

Open Geospatial Consortium (Ed.), 2000. The OpenGIS Abstract Specification, Topic 6: The Coverage Type and its Subtypes. Version 4, doc.no. 00-106, OGC.

Persson, D., Gartner, G., Buchroithner, M., 2006. Towards a Typology of Interactivity Functions for Visual Map Exploration. In: Stefanakis, E., Peterson, M. P., Armenakis, C., Delis, V. (Eds.), Geographic Hypermedia. Springer, Ch. 15, pp. 275–292.

Rost, R. J., January 2006. OpenGL(R) Shading Language (2nd Edition). Addison-Wesley Professional.

Spence, R., 2000. Information Visualization. Addison Wesley.

Tanner, C. C., Migdal, C. J., Jones, M. T., 1998. The clipmap: a virtual mipmap. In: SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM, New York, NY, USA, pp. 151–158.

Vretanos, P. A. (Ed.), 2005a. OpenGIS Filter Encoding Implementation Specification. Version 1.1.0, doc.no. 04-095, OGC.

Vretanos, P. A. (Ed.), 2005b. Web Feature Service Implementation Specification. Version 1.1.0, doc.no. 04-094, OGC.

Šúri, M., Hofierka, J., 2004. A New GIS-based Solar Radiation Model and Its Application to Photovoltaic Assessments. Transactions in GIS 8 (2), 175–190.

Williamson, C., Shneiderman, B., 1992. The dynamic homefinder: evaluating dynamic queries in a real-estate information exploration system. In: SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on research and development in information retrieval. ACM, New York, NY, USA, pp. 338–346.

Wood, J., Kirschenbauer, S., Döllner, J., Lopes, A., Bodum, L., 2005. Using 3D in Visualization. In: Dykes, J., MacEachren, A., Kraak, M.-J. (Eds.), Exploring Geovisualization. Elsevier, London, Ch. 14, pp. 295–312.

Yi, J., ah Kang, Y., Stasko, J., Jacko, J., 2007. Toward a deeper understanding of the role of interaction in information visualization. IEEE Transactions on Visualization and Computer Graphics 13 (6), 1224–1231.